ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational and Applied Mathematics 183 (2005) 245–258

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

A MATLAB implementation of upwind finite differences and adaptive grids in the method of lines

A. Vande Wouwer[a,*], P. Saucez[b], W.E. Schiesser[c], S. Thompson[d]

[a]*Faculté Polytechnique de Mons, Service d'Automatique, Boulevard Dolez 31, Mons 7000, Belgium*
[b]*Faculté Polytechnique de Mons, Service de Mathématique et Recherche Opérationnelle, Belgium*
[c]*Mathematics and Engineering, Lehigh University, Bethlehem, USA*
[d]*Department of Mathematics and Statistics, Radford University, USA*

Received 20 May 2004; received in revised form 6 September 2004

Abstract

In this paper, we report on the development of a MATLAB library for the solution of partial differential equation systems following the method of lines. In particular, we focus attention on upwind finite difference schemes and grid adaptivity, i.e., grid movement or grid refinement. Several algorithms are presented and their performance is demonstrated with illustrative examples including a fixed-bed reactor with periodic flow reversal, a model of flame propagation, and the Korteweg–de Vries equation.

MSC: 65M20; 65M06; 65M50

Keywords: Partial differential equations; Grid refinement; Moving grid; Catalytic combustion; Flame propagation; Korteweg–de Vries equation

1. Introduction

Computational modeling is now routinely applied in various disciplines of science and engineering. As the systems under consideration are often characterized by several independent variables, e.g., space and time, they are described by sets of, generally nonlinear, partial differential equations (PDEs). One

* Corresponding author. Tel.: +32 6537 4141; fax: +32 6537 4136.
E-mail address: Alain.VandeWouwer@fpms.ac.be (A. Vande Wouwer).

0377-0427/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2004.12.030

of the most popular approaches to the numerical solution of PDE models is the method of lines (MOL), which proceeds in two separate steps:

- approximation of the spatial derivatives using finite difference, finite element or finite volume techniques;
- time integration of the resulting semi-discrete (discrete in space, but continuous in time) equations using an appropriate solver.

The success of the MOL stems from its simplicity of implementation and the availability of high-quality time integrators for solving a wide range of problems, including ordinary differential equations (ODEs), and mixed systems of algebraic and ordinary differential equations (AEs and ODEs forming a system of differential-algebraic equations (DAEs)).

Several general-purpose FORTRAN libraries, e.g., NAG [1,8] or DSS/2 [12], can be used to develop codes following the MOL approach. Recently, several MATLAB-based libraries have also been proposed for the solution of ODE/DAE systems [13] and for the solution of PDE systems using spectral methods [18,14]. MATLAB is now widely available in industry and academia and provides a very convenient basis for the development of MOL tools, allowing compact vector/matrix operations, and requiring minimum programming expertise.

In a recent paper [16], the authors report on the development of a collection of MATLAB routines (called MATMOL) implementing various finite difference schemes (FDs) and flux limiters, as well as a discussion of some preliminary results concerning the implementation of a grid refinement strategy. The present paper elaborates on these preliminary results and focuses attention on the following techniques applied to PDE problems with solutions displaying moving fronts (e.g., moving fronts of temperature and concentration, water waves):

- upwind finite difference schemes for the solution of convection–diffusion–reaction problems, with application to a catalytic fixed-bed reactor operated with periodic flow reversal;
- a dynamic grid adaptation strategy based on the equidistribution principle and ideas borrowed from [17,2], with application to a flame propagation problem and Korteweg–de Vries equation;
- a more elaborate version of our static grid refinement strategy, with application to the same two standard problems.

This paper is organized as follows. The next section briefly introduces the MOL strategy and the development of MATLAB routines for spatial discretization. Section 3 presents upwind finite difference schemes and their application to a catalytic combustion problem [4]. In Section 4, the MATLAB implementation of a moving grid algorithm, similar in spirit to the FORTRAN code MOVGRD [17,2], is discussed. The algorithm is then applied to two standard PDE problems, i.e., Dwyer and Sanders' flame propagation problem [3], and the classical Korteweg–de Vries equation [7]. Section 5 presents the MATLAB implementation of a static grid refinement algorithm, based on the FORTRAN code AGEREG [10], and its application to the two standard PDE problems considered in the previous section. Finally, Section 6 draws some conclusions.

2. A MATLAB-based method of lines library

Consider the PDE problem

$$x_t = f(z, t, x, x_z, x_{zz}, x_{zzz}, \ldots), \quad z \in \Omega, \quad t \geqslant 0, \tag{1}$$

$$0 = b(z, t, x, x_z, \ldots), \quad z \in \Gamma, \quad t > 0, \tag{2}$$

$$x(t = 0, z) = x_0(z), \quad z \in \Omega \cup \Gamma, \tag{3}$$

where $x \in \Re^{n_{\text{pde}}}$ is the vector of dependent variables (e.g., temperature, concentration), z is the vector of spatial independent variables (in general, three orthogonal spatial coordinates), and t is an initial value independent variable (typically time). A subscript notation is used for the several partial derivatives, i.e., $x_t = \partial x/\partial t$, $x_z = \partial x/\partial z$. Eqs. (1)–(3) represent a system of PDEs defined in a spatial domain Ω, their associated boundary conditions (BCs) defined on the boundary surface Γ of Ω, and initial conditions (ICs) defined on the complete spatial domain.

Following the MOL, finite differences (or other techniques such as spectral methods, but in the continuation of this article we will restrict ourselves to FDs) can be used to approximate the spatial derivatives appearing in PDEs (1) and BCs (2) and the corresponding linear transformation can be conveniently implemented using the concept of a differentiation matrix D, i.e.,

$$\tilde{x}_z = D_1\tilde{x}, \tag{4}$$

$$\tilde{x}_{zz} = D_2\tilde{x}, \tag{5}$$

$$\vdots\,. \tag{6}$$

Substitution of (4)–(6) into (1)–(2) yields a semi-discrete ODE or DAE system

$$\tilde{x}_t = f(z, t, \tilde{x}, \tilde{x}_z, \tilde{x}_{zz}, \tilde{x}_{zzz}, \ldots), \quad z \in \Omega, \quad t \geqslant 0, \tag{7}$$

$$0 = b(z, t, \tilde{x}, \tilde{x}_z, \ldots), \quad z \in \Gamma, \quad t > 0, \tag{8}$$

$$\tilde{x}(t = 0, z) = x_0(z), \quad z \in \Omega \cup \Gamma, \tag{9}$$

where $\tilde{x}$ is the approximate solution. This ODE/DAE system can be integrated in time using one of the solvers available in the MATLAB ODE Suite [13], e.g., ode15s (which is suitable for stiff ODEs and index 1 DAEs).

3. Upwinding in the method of lines

For convective PDE problems, upwind FDs [9], e.g.,

$$\tilde{x}_z(z_i) = (-x(z_{i-3}) + 6x(z_{i-2}) - 18x(z_{i-1}) + 10x(z_i) + 3x(z_{i+1}))/(12\Delta z) \tag{10}$$

for a flow from left to right or

$$\tilde{x}_z(z_i) = (-3x(z_{i-1}) - 10x(z_i) + 18x(z_{i+1}) - 6x(z_{i+2}) + x(z_{i+3}))/(12\Delta z) \tag{11}$$

for a flow from right to left, work very effectively and avoid spurious oscillations as generated by centered FDs. Upwind schemes with various orders of accuracy have been implemented in MATLAB, either on uniform grids or on nonuniform grids (to this end, the algorithm WEIGHTS of Fornberg [5] can be very conveniently used to compute the finite difference weights).

For illustration purposes, the five-point (fourth-order accurate) biased upwind (10) and (11) on uniform grids are now applied to a catalytic combustion problem described in [4].

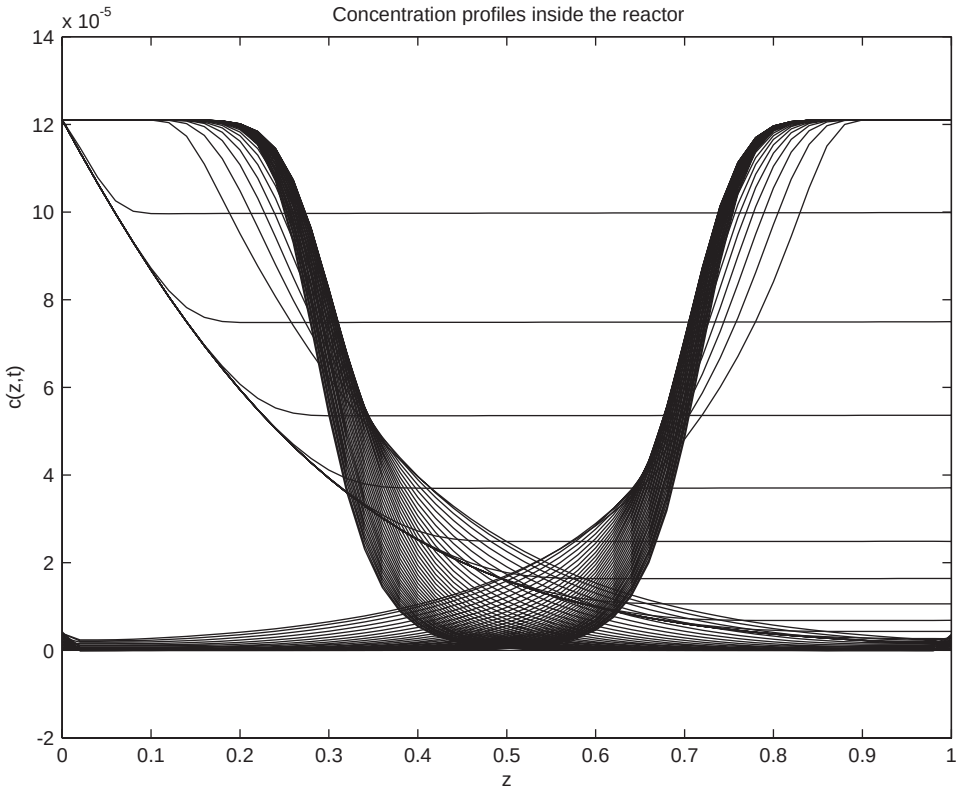

Fig. 1. Evolution of the concentration profiles in the catalytic fixed-bed reactor at $t = 0, 100, \ldots, 9500$.

Mass and energy balances yield the following PDEs:

$$c_t = Dc_{zz} - vc_z - kc\mathrm{e}^{(-E/RT)},$$
$$\rho\bar{c}_p T_t = \lambda T_{zz} - \rho_{\mathrm{g}} c_{p\mathrm{g}} \varepsilon T_z + \varepsilon kc\mathrm{e}^{(-E/RT)}(-\Delta H_{\mathrm{R}}), \quad 0 < z < L, \quad t > 0, \tag{12}$$

where c (kmol/m^3) is the concentration of reactive species, T (K) the temperature, $D = 5 \times 10^{-3}$ m^2/s the diffusivity constant, $v = 1$ m/s the gas velocity, $k = 29\,732$ s^{-1} the rate constant, $E/R = 8000$ K the activation temperature, $\rho\bar{c}_p = 400$ kJ/(m^3 K) the heat capacity of the fixed bed, $\lambda = 2.06 \times 10^{-3}$ kW/(mK) the effective axial heat conductivity, $\rho_{\mathrm{g}} c_{p\mathrm{g}} = 0.5$ kJ/(m^3 K) the heat capacity of the gas, $\varepsilon = 0.8$ the bed void fraction, $(-\Delta H_{\mathrm{R}}) = 206\,000$ kJ/kmol the heat of reaction, $L = 1$ m the reactor length.

For weakly exothermic reactions, operation of the fixed-bed catalytic reactor with periodic flow reversals is of particular interest. This way, the front and end parts of the catalyst bed act as regenerative heat exchangers for feed and effluent, allowing the reactions to be operated autothermally at high temperatures.

After a start-up phase (here $t_{\mathrm{start}} = 1500$ s), where the gas enters at high temperature ($c_{\mathrm{in}} = 1.21 \times 10^{-4}$ kmol/m^3 at $T_{\mathrm{in}} = 873$ K), the feed temperature is decreased ($T_{\mathrm{in}} = 293$ K) and periodic flow reversal allows the reactor to be operated autothermally.

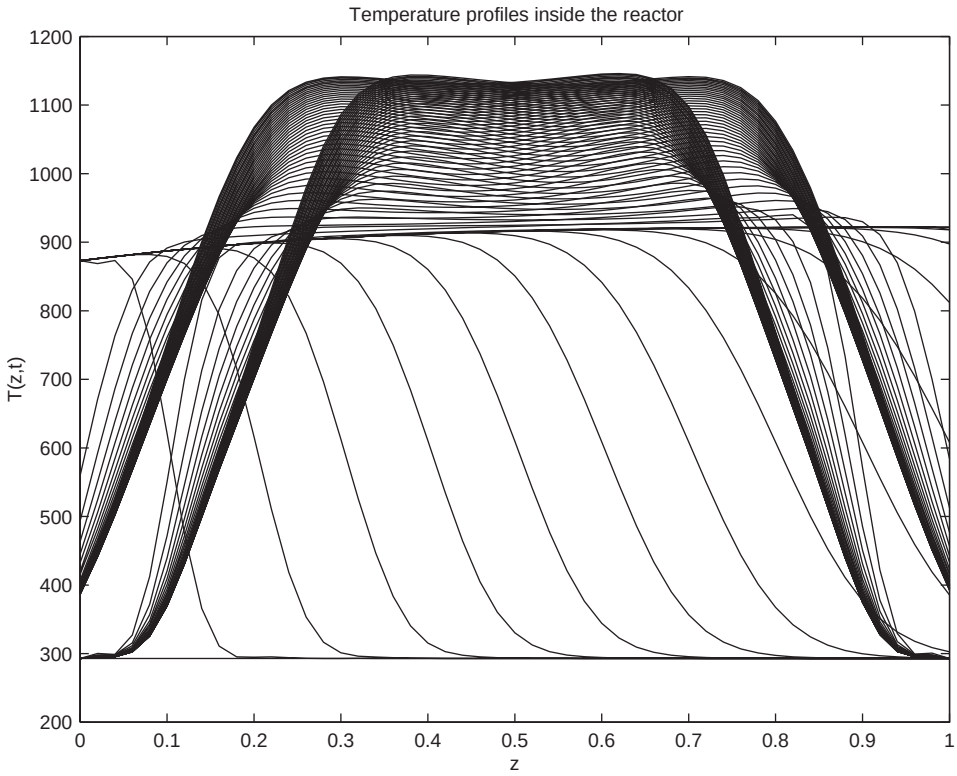

Fig. 2. Evolution of the temperature profiles in the catalytic fixed-bed reactor at $t = 0, 100, \ldots, 9500$.

From a modeling point of view, periodic flow reversal involves periodic changes in the boundary conditions, i.e. (for $k = 0, 2, 4, \ldots$)

$$c(z = 0, t) = c_{\text{in}}(t), \quad T(z = 0, t) = T_{\text{in}}(t), \quad k\Delta\tau \leqslant t < (k + 1)\Delta\tau, \tag{13}$$

$$c(z = L, t) = c_{\text{in}}(t), \quad T(z = L, t) = T_{\text{in}}(t), \quad (k + 1)\Delta\tau \leqslant t < (k + 2)\Delta\tau \tag{14}$$

and periodic changes in the sign of the gas velocity v. From a numerical point of view, this requires the use of appropriate differentiation matrices based on either (10) and (11). Figs. 1 and 2 show the evolution of the concentration and temperature at $t = 0, 100, \ldots, 9500$. This solution can be computed on a fixed uniform grid with a relatively small number of grid points, e.g., $N = 51$, as the moving fronts remain quite smooth. Time integration is performed using the solver ode15s with $\text{RelTol} = 10^{-3}$ and $\text{AbsTol} = 10^{-6}$. The elements of the concentration and temperature vectors are interlaced so as to confer a banded structure to the Jacobian matrix, which can be specified using a function JPattern.

4. A dynamic grid adaptation algorithm

When considering PDE problems with steep moving fronts, it can be advantageous to concentrate the grid points in spatial regions of interest and to move them continuously in time, i.e., to use dynamic grid

adaptation, so that their locations follow the moving fronts and remain near optimal. Not only the number of grid points can be significantly reduced, as fewer grid points are used in regions of low solution activity, but also larger time steps can usually be taken as the moving fronts are less likely to cross grid points (generating large changes in time derivatives at these grid points).

In this section, we consider the development of dynamic grid adaptation algorithm based on ideas borrowed from [17]. This algorithm is based on the Lagrangian formulation of the PDE problem (1). Consider the continuous time trajectories of the grid points

$$z_1 < z_2(t) < \cdots < z_i(t) < \cdots < z_N. \tag{15}$$

Along $z(t) = z_i(t)$ the total temporal derivative of x is given by

$$\dot{x} = x_t + \dot{z}x_z = f(x) + \dot{z}x_z. \tag{16}$$

The ODEs defining the grid point movement, i.e., $\dot{z} = g(t)$, can be derived based on some physical a priori knowledge, such as a flow-related quantity, or so as to equally distribute a monitor function $m(x)$ such as the arc-length of the solution (several other monitor functions can be considered, e.g., based on the solution curvature), i.e.,

$$m(x) = \sqrt{(\alpha + \|u_z\|_2^2)}, \tag{17}$$

$$M_{i-1}\Delta z_{i-1} = M_i \Delta z_i = c, \quad 2 \leqslant i \leqslant N-1, \tag{18}$$

where $\Delta z_i = z_{i+1} - z_i$ is the local grid spacing, M_i is a discrete approximant of the monitor function $m(x)$ in the grid interval $[z_i, z_{i+1}]$, and c is a constant. In order to avoid excessive spatial distortion and temporal oscillation of the grid, two regularization procedures are used in [17]. To this end, the spatial equidistribution equation (18) is expressed in terms of the grid density $n_i = 1/\Delta z_i$

$$\frac{n_{i-1}}{M_{i-1}} = \frac{n_i}{M_i}, \quad 2 \leqslant i \leqslant N-1. \tag{19}$$

First, spatial smoothing is accomplished by replacing the grid density n_i in (19) by

$$\begin{aligned}
&\tilde{n}_0 = n_0 - \kappa(\kappa+1)(n_1 - n_0),\\
&\tilde{n}_i = n_i - \kappa(\kappa+1)(n_{i+1} - 2n_i + n_{i-1}), \quad 2 \leqslant i \leqslant N-1,\\
&\tilde{n}_N = n_N - \kappa(\kappa+1)(n_{N-1} - n_N),
\end{aligned} \tag{20}$$

where κ is a positive parameter. The introduction of the 'anti-diffused' density $\tilde{n}_i$ ensures that the grid is locally bounded, i.e., that adjacent grid spacings do not differ too much from one another (the complete developments can be found in [17])

$$\frac{\kappa}{\kappa+1} \leqslant \frac{n_{i-1}}{n_i} \leqslant \frac{\kappa+1}{\kappa}. \tag{21}$$

Second, temporal smoothing is accomplished by replacing the system of AE (19) by a system of differential equations

$$\frac{\tilde{n}_{i-1} + \tau\dot{\tilde{n}}_{i-1}}{M_{i-1}} = \frac{\tilde{n}_i + \tau\dot{\tilde{n}}_i}{M_i}, \quad 2 \leqslant i \leqslant N-1, \tag{22}$$

where the positive parameter τ acts as a time constant preventing abrupt changes in the grid movement.

Experience shows that spatial smoothing is more important than temporal smoothing.

In contrast with the original Fortran software implementation MOVGRD [2], in which a nonlinear Galerkin discretization of the Lagrangian description of the PDEs (16) is applied, our MATLAB implementation is based on simple finite difference schemes on nonuniform grids (as introduced in the previous section). This choice has the advantage of simplicity and flexibility. Indeed, finite difference schemes can accommodate any spatial differential operators, whereas the nonlinear Galerkin discretization used in MOVGRD is dedicated to convection–diffusion problems. However, finite difference schemes can be less efficient on some of these problems.

The semi-discrete equations resulting from (16) are then combined with Eq. (22) to yield a system of ODEs in the form of

$$\begin{aligned} &\dot{x} - A(x,z)\dot{z} = \tilde{f}(x,z), \\ &\tau B(x,z)\dot{z} = g(x,z) \end{aligned} \tag{23}$$

or

$$M(\xi)\dot{\xi} = q(\xi), \tag{24}$$

where ξ is a vector of dependent variables in which the elements of x and z are interlaced (following the progression of the spatial grid), so that M takes a block-pentadiagonal matrix structure (the dimension of the blocks is $n_{\text{pde}}+1 \times n_{\text{pde}}+1$). The ODE system (24) is solved using the BDF/NDF solver ode15s [13]. The sparsity pattern of the mass matrix and of the Jacobian matrix is defined using functions MvPattern and JPattern.

To illustrate the performance of this code, we consider first a model of flame propagation [3] consisting of two coupled equations for mass density and temperature

$$\begin{aligned} &\rho_t = \rho_{zz} - N_{DA}\rho, \\ &T_t = T_{zz} + N_{DA}\rho, \quad 0 < z < 1, \quad t > 0, \end{aligned} \tag{25}$$

where $N_{DA} = 3.52 \times 10^6 \ \mathrm{e}^{-4/T}$.

The initial conditions are given by

$$\rho(z,0) = 1, \quad T(z,0) = 0.2, \quad 0 \leqslant z \leqslant 1, \tag{26}$$

and the boundary conditions are

$$\begin{aligned} &\rho_z(0,t) = 0, \quad T_z(0,t) = 0, \\ &\rho_z(1,t) = 0, \quad T(1,t) = f(t), \quad t \geqslant 0, \end{aligned} \tag{27}$$

with

$$f(t) = \begin{cases} 0.2 + t/2 \times 10^{-4}, & t \leqslant 2 \times 10^{-4}, \\ 1.2, & t \geqslant 2 \times 10^{-4}. \end{cases} \tag{28}$$

The heat source located at $z = 1$ generates a flame front that propagates from right to left at an almost constant speed.

From a numerical point of view, the main challenge in this problem is to accurately capture the ignition phase and to subsequently reproduce the correct propagation speed of the flame front. To this end, a relatively large number of points, e.g., $N = 501$, is needed when using finite differences on fixed uniform

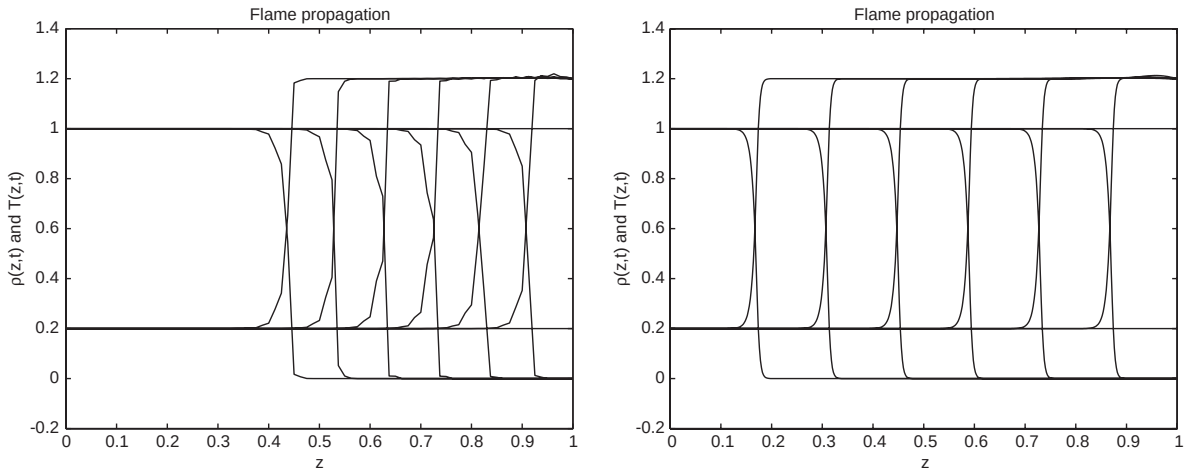

Fig. 3. Fixed uniform grid solution of the flame propagation problem at $t = 0.001, \ldots, 0.006$. The right plot corresponds to $N = 81$ grid points, whereas the left plot corresponds to $N = 501$ gridpoints. The propagation speed is largely underestimated in the former case.

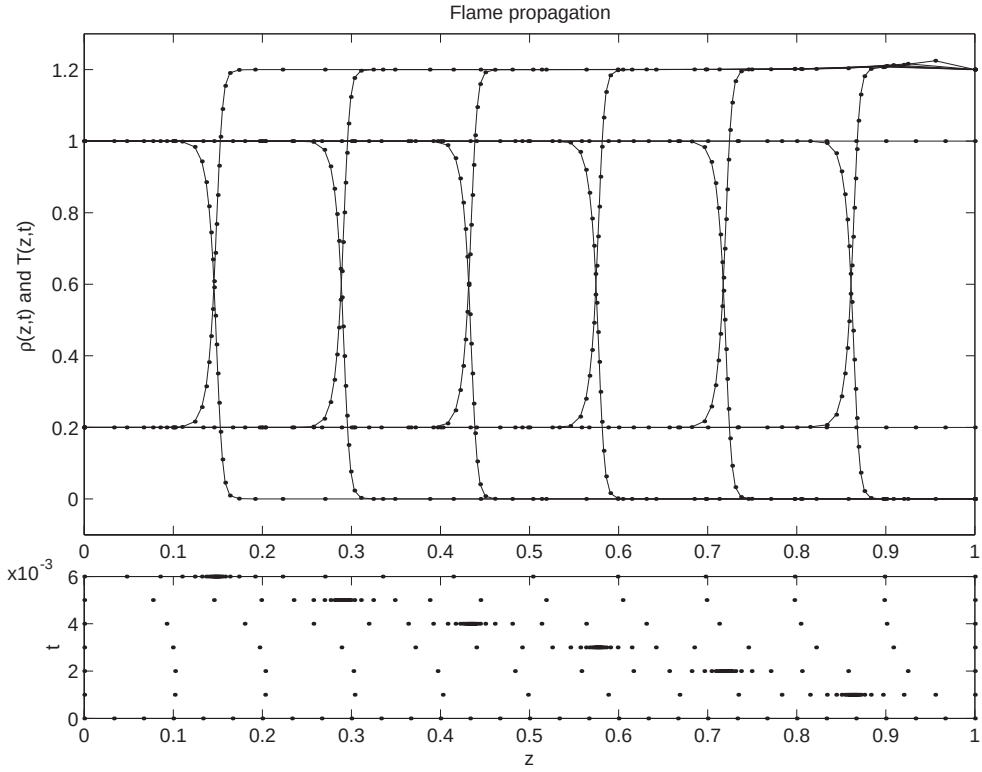

Fig. 4. Moving grid solution of the flame propagation problem: $N = 29$, $\alpha = 0.5$, $\kappa = 1$, $\tau = 10^{-6}$.

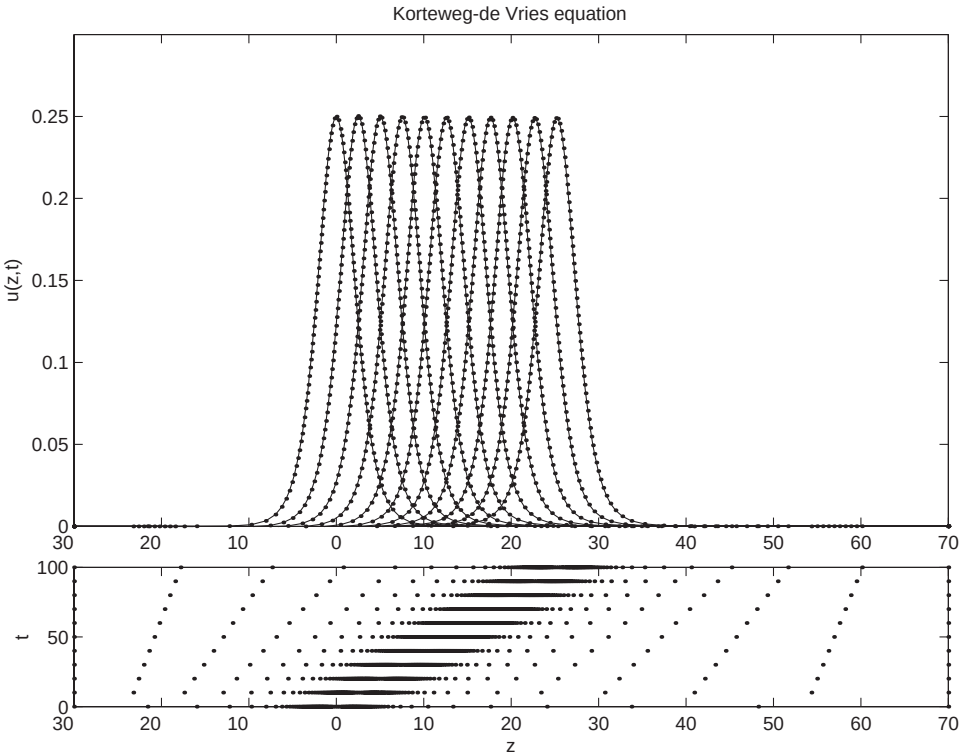

Fig. 5. Moving grid solution of the Korteweg–de Vries equation: initialisation with grid refinement, $N = 113$, $\alpha = 0.0$, $\kappa = 2$, $\tau = 10^{-4}$ (solid line: analytic solution).

grids. Fig. 3 illustrates two extreme cases corresponding to $N = 81$ and 501, respectively. In the former case, the propagation speed is largely underestimated. The problem is solved on the time interval (0, 0.006) and time integration is performed using ode15s with AbsTol $= 10^{-3}$ and RelTol $= 10^{-3}$. A moving grid, which concentrates the nodes where they are needed, could therefore be advantageous in this case. The following tuning parameter values are selected: $N = 29$, $\alpha = 0.5$, $\kappa = 1$, $\tau = 10^{-6}$. Time integration is performed using ode15s with AbsTol $= 10^{-3}$ and RelTol $= 10^{-3}$. Very satisfactory numerical results are obtained, which are represented in Fig. 4. The computational expense is about the same as the one required to compute a fixed grid solution with $N = 81$ (see Fig. 3—about 15 s with an AMDAthlon2400+), and represents about $\frac{1}{30}$ of that required with $N = 501$.

As a second test example, we consider the classical Korteweg–de Vries equation, which was originally introduced by Korteweg and de Vries in 1895 [7] to describe the behavior of small amplitude shallow-water waves in one space dimension,

$$u_t = -6uu_z - u_{zzz} \quad -\infty \leqslant z \leqslant \infty, \quad t \geqslant 0, \tag{29}$$

$$= -3(u^2)_z - u_{zzz}, \tag{30}$$

$$u(z, 0) = u_0(z), \tag{31}$$

which combines the effect of nonlinearity and dispersion. In the following, the propagation of a single soliton

$$u(z,t) = 0.5s\,\mathrm{sech}^2\,[0.5\sqrt{s}(z-st)] \tag{32}$$

is investigated numerically. $u(z,t)$ of (32) is the analytical solution to (30)–(31) and serves as a standard by which the accuracy of the numerical solution can be assessed. Particular attention must be paid to the selection of finite difference schemes for the approximation of the spatial derivatives. Here, the second form (30) of the equation is used. The first-order derivative term is computed using a five-point biased upwind scheme, and the third-order derivative term is computed using stagewise differentiation, i.e., $\tilde{u}_{zzz}=D_1(D_1(D_1\tilde{u}))$, with a three-point centered differentiation matrix D_1. Fig. 5 shows the propagation of a single soliton on the time interval [0, 50]. Time integration is performed using ode15s with AbsTol$=10^{-3}$ and RelTol $= 10^{-3}$. The following tuning parameter values are selected: $\alpha = 0.0$, $\kappa = 2$, $\tau = 10^{-4}$. To determine an appropriate number of grid points and to initially concentrate them in the soliton, a call to the grid refinement algorithm presented in the next section (with $c = 0.005$ and $K = (\kappa+1)/\kappa$) is made at $t = 0$. As a result, $N = 113$ grid points are used, which very effectively follow the soliton in its movement. The computational expense is reasonable, e.g., about 115 s with an AMDAthlon2400+.

5. A static grid refinement algorithm

An alternative procedure to adapt the grid point locations is static grid refinement. Basically, this approach proceeds in four steps:

(1) approximation of the spatial derivatives on a fixed nonuniform grid;
(2) time integration of the resulting semi-discrete ODEs;
(3) adaptation/refinement of the spatial grid;
(4) interpolation of the solution to produce new initial conditions.

The movement of the grid points is therefore no longer continuous in time, but time integration is halted periodically to update the grid. This approach has several advantages:

- it is simple to implement;
- PDE solution and grid adaptation/refinement are uncoupled and can be programmed in separate MATLAB functions (programming therefore requires less expertise from the end user);
- there are N ODEs to solve (for the PDE solution) instead of $2N$ ODEs (for the PDE solution and the grid movement in the previous approach);
- grid refinement allows spatial accuracy to be controlled as grid points are added or deleted depending on the evolution of spatial activity of the solution (for instance the birth of new moving fronts can be easily accommodated, whereas a dynamic moving grid with a fixed number of grid points can be unable to resolve unforeseen spatial activity with enough accuracy).

But has also several drawbacks:

- as the grid movement is no longer continuous in time, grid points are suboptimally located, i.e., they can for instance lag behind the front if the grid is not updated sufficiently frequently;

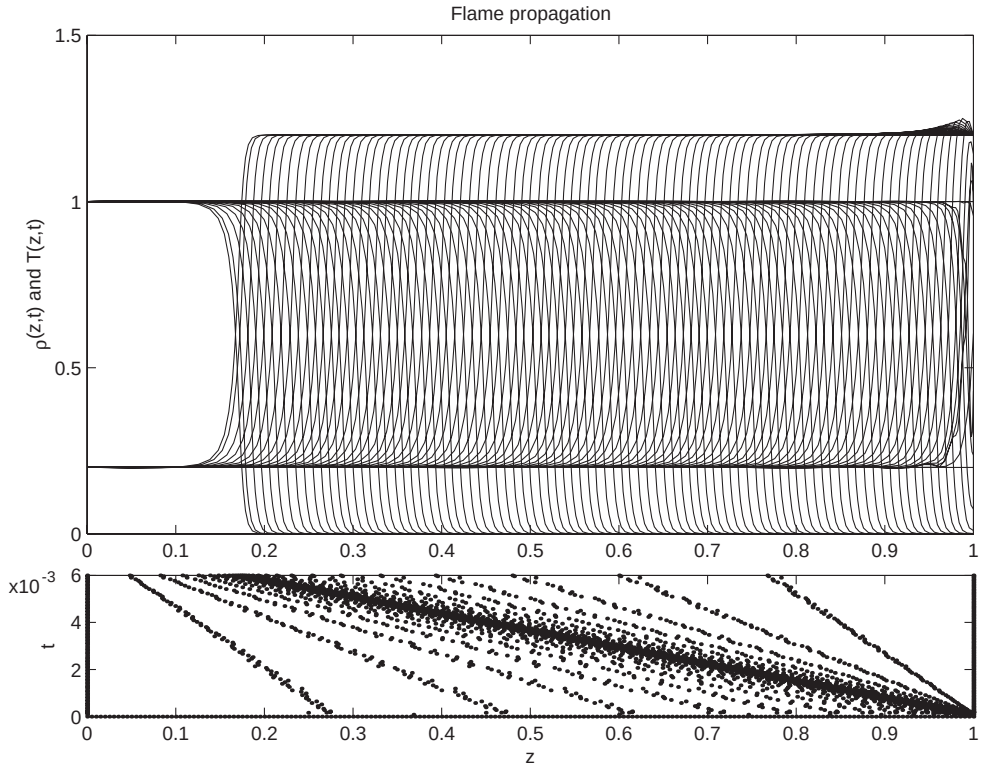

Fig. 6. Adaptive grid refinement solution of the flame propagation problem: $\bar{N} = 33$, $\alpha = 0.05$, $c = 0.2$, $K = 1.4$.

- periodic solver restarts involve some computational overhead (and usually one-step solvers, such as implicit Runge–Kutta or Rosenbrock algorithms, which do not require the past solution history, will perform better than BDF solvers);
- interpolation to produce new initial conditions is an additional source of errors.

Despite these drawbacks, the authors have successfully applied an adaptive grid refinement algorithm, called AGEREG [15], to a variety of problems (see also [11] in this journal issue for numerical experiments with the original Fortran code and a generalized fifth-order KdV equation), and a MATLAB implementation is considered in the following.

Again, the grid movement is based on the equidistribution of a specified monitor function (18). To limit grid distortion, a spatial regularization procedure due to Kautsky and Nichols [6] is used, which ensures that the grid is locally bounded with respect to a constraint $K \geqslant 1$ (see also (21), where $K = (\kappa + 1)/\kappa$),

$$\frac{1}{K} \leqslant \frac{\Delta z_i}{\Delta z_{i-1}} \leqslant K, \quad 2 \leqslant i \leqslant N - 1. \tag{33}$$

To ensure this property, the monitor function is modified, keeping its maximal values, but increasing its minimum values, in a procedure called “padding”. The resulting padded monitor function is then equidistributed yielding a grid whose ratios of consecutive grid steps are bounded as required. In practice, the padding is chosen (there is, in principle, an infinity of possibilities to achieve padding) so that the

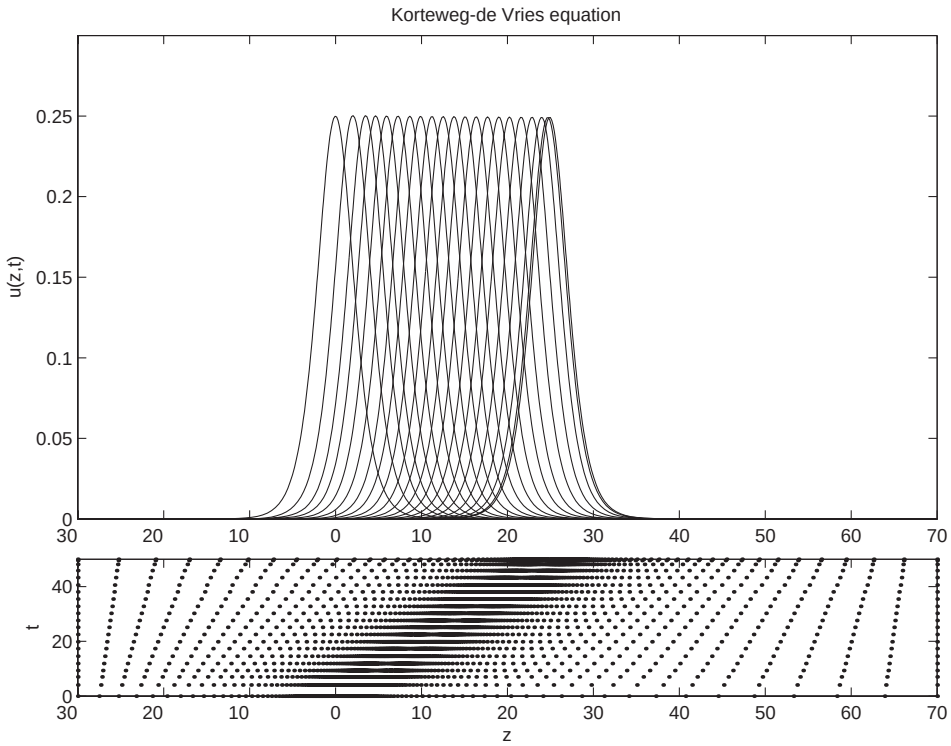

Fig. 7. Adaptive grid refinement solution of the Korteweg–de Vries equation: $\bar{N} = 154$, $\alpha = 0.0$, $c = 0.2$, $K = 1.4$.

equidistributing grid has adjacent steps with constant ratios equal to the maximum allowed. MATLAB implementation involves the following issues:

- the Rosenbrock solver ode23s is usually the preferred algorithm as it does not require the past solution history;
- the solver is periodically halted, after a fixed number of time steps, using an *Events* function, which monitors the evolution of the number of steps;
- the solution is interpolated using cubic splines as implemented in *Spline* in order to generate initial conditions on the new grid.

To evaluate the performance of this code, we consider the same two test examples, e.g., the flame propagation problem (25), and the classical Korteweg–de Vries (29)–(30).

Fig. 6 shows the solution of the flame propagation problem (25), using a curvature-based monitor function

$$m(u) = \sqrt{(\alpha + \|u_{xx}\|_\infty} \tag{34}$$

and the tuning parameters $\alpha = 0.05$, $c = 0.2$, $K = 1.4$, AbsTol $= 10^{-3}$, and RelTol $= 10^{-3}$. In this figure, the solution is plotted every $n_{\text{steps}} = 10$, when ode23s is halted to refine the grid, in order to show how the adaptive grid solution proceeds. The average number of grid points is $\bar{N} = 33$, and the

computational expense is similar to the one required with the moving grid algorithm (i.e., about 15 s with an AMDAthlon2400+). The grid refinement algorithm is particularly robust with respect to parameter tuning, and numerical solutions can be computed according to the desired level of resolution. Fig. 7 shows the propagation of a single soliton. Here, an arc-length monitor function (17) is used, and the following tuning parameters are selected $\alpha = 0.0$, $c = 0.2$, $K = 1.4$, AbsTol $= 10^{-4}$, and RelTol $= 10^{-3}$. The solution is plotted every $n_{\text{steps}} = 1$, i.e., ode23s is halted every time step to refine the grid. The algorithm is extremely effective, generating a solution in only 2.5 s with an AMDAthlon2400+! The average number of grid points is $\bar{N} = 154$.

6. Conclusions

In this paper, we report on the development of a MATLAB library for the method of lines solution of partial differential equation problems. Particularly, we focus attention on PDE problems with steep moving fronts, and the use of upwind finite differences and grid adaptation/refinement. Grid adaptivity appears to be a very useful tool, allowing the computational load to be significantly reduced, and small-scale features of the solution to be captured with better accuracy (than fixed grid with a comparable number of points). In addition, grid refinement allows spatial accuracy to be controlled, and problems with time-varying spatial activity (birth or decay of fronts) to be accommodated. A MATLAB implementation of AGEREG [15] is presented, which performs very well on nonlinear wave propagation problems. The toolbox is available on request from the authors (Alain.VandeWouwer@fpms.ac.be or wes1@lehigh.edu) and also from the web at www.autom.fpms.ac.be

References

[1] M. Berzins, Developments in NAG library software for parabolic equations, in: J.C. Mason (Ed.), Scientific Software Systems, Chapman & Hall, London, 1989.
[2] J.G. Blom, P.A. Zegeling, Algorithm 731: a moving-grid interface for systems of one-dimensional time-dependent partial differential equations, ACM Trans. Math. Software 20 (1994) 194–214.
[3] H.A. Dwyer, B.R. Sanders, Numerical modeling of unsteady flame propagation, Sandia National Laboratory Livermore Report SAND77-8275, 1978.
[4] G. Eigenberger, C. Nieken, Catalytic combustion with periodic flow reversal, Chem. Eng. Sci. 43 (1988) 2109–2115.
[5] B. Fornberg, Calculation of weights in finite difference formulas, SIAM Rev. 40 (1998) 685–691.
[6] J. Kautsky, N.K. Nichols, Equidistributing meshes with constraints, SIAM J. Sci. Stat. Comput. 1 (1980) 499–511.
[7] D.J. Korteweg, G. de Vries, On the change of form of long waves advancing in a rectangular canal and on a new type of long stationary waves, Philos. Mag. 39 (1895) 422–443.
[8] S.V. Pennington, M. Berzins, New NAG library software for first-order systems of time-dependent PDEs, ACM Trans. Math. Software 20 (1994) 63.
[9] P. Saucez, W.E. Schiesser, A. Vande Wouwer, Upwinding in the method of lines, Math. Comput. Simulat. 56 (2001) 171–185.
[10] P. Saucez, A. Vande Wouwer, W.E. Schiesser, Some observations on a static spatial remeshing method based on equidistribution principles, J. Comput. Phys. 128 (1996) 274–288.
[11] P. Saucez, A. Vande Wouwer, P. Zegeling, Adaptive mesh computations for the extended fifth-order Korteweg–de Vries equation, J. Comput. Appl. Math., this issue, doi:10.1016/j.cam.2004.12.028.
[12] W.E. Schiesser, The Numerical Method of Lines: Integration of Partial Differential Equations, Academic Press, San Diego, 1991.
[13] L.F. Shampine, I. Gladwell, S. Thompson, Solving ODEs with MATLAB, Cambridge University Press, Cambridge, 2003.

[14] L.N. Trefethen, Spectral Methods in MATLAB, SIAM, Philadelphia, 2000.
[15] A. Vande Wouwer, Ph. Saucez, W.E. Schiesser (Eds.), Adaptive Method of Lines, Chapman & Hall/CRC, Boca Raton, 2001.
[16] A. Vande Wouwer, Ph. Saucez, W.E. Schiesser, Simulation of distributed parameter systems using a MATLAB-based method of lines toolbox—chemical engineering applications, Ind. Eng. Chem. Res. 43 (2004) 3469–3477.
[17] J.G. Verwer, J.G. Blom, R.M. Furzeland, P.A. Zegeling, A moving-grid method for one-dimensional PDEs based on the method of lines, in: J.E. Flaherty, P.J. Paslow, M.S. Shephard, J.D. Vasilakis (Eds.), Adaptive Methods for Partial Differential Equations, SIAM, Philadelphia, 1989, pp. 160–175.
[18] J.A.C. Weideman, S.C. Reddy, A MATLAB differentiation matrix suite, ACM Trans. Math. Software 26 (2000) 465–519.